US009436572B2

(12) **United States Patent**
Choi et al.

(10) **Patent No.: US 9,436,572 B2**
(45) **Date of Patent: Sep. 6, 2016**

(54) **METHOD AND SYSTEM TO DETECT A CACHED WEB PAGE**

(71) Applicant: **eBay Inc.**, San Jose, CA (US)

(72) Inventors: **Gregory Choi**, San Jose, CA (US); **Diego Lagunas**, San Jose, CA (US); **Sathishwar Pottavathini**, Dublin, CA (US)

(73) Assignee: **eBay Inc.**, San Jose, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **14/293,925**

(22) Filed: **Jun. 2, 2014**

(65) **Prior Publication Data**

US 2014/0289317 A1 Sep. 25, 2014

Related U.S. Application Data

(63) Continuation of application No. 12/176,253, filed on Jul. 18, 2008, now Pat. No. 8,745,164.

(60) Provisional application No. 60/950,774, filed on Jul. 19, 2007.

(51) **Int. Cl.**
G06F 15/16 (2006.01)
G06F 11/30 (2006.01)
(Continued)

(52) **U.S. Cl.**
CPC ***G06F 11/3037*** (2013.01); ***G06F 12/0815*** (2013.01); ***G06F 17/30902*** (2013.01); ***H04L 67/2842*** (2013.01)

(58) **Field of Classification Search**
CPC H04L 67/2842
USPC 709/203
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,821,927 A 10/1998 Gong
5,924,116 A * 7/1999 Aggarwal G06F 17/30902 707/E17.12

(Continued)

FOREIGN PATENT DOCUMENTS

WO WO-2006033850 A2 3/2006
WO WO-2006033850 A3 3/2006

(Continued)

OTHER PUBLICATIONS

Davis et al., Formal Models for Web Navigation with Session Control and Browser Cache, 2004, ICFEM 2004, LNCS 3308, pp. 46-60.*

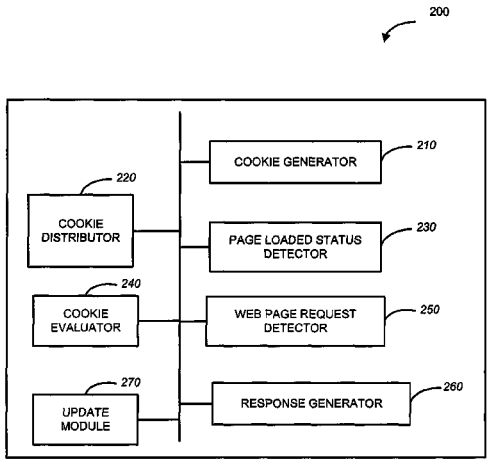
(Continued)

Primary Examiner — Peter Shaw
(74) *Attorney, Agent, or Firm* — Schwegman, Lundberg & Woessner, P.A.

(57) **ABSTRACT**

A method and system to determine whether a web page has been cached is provided. An example system comprises a cookie generator, a cookie distributor, and a cookie evaluator. The cookie distributor may be configured to provide the code to a client system, in response to a request for web content from the client system. A value of the code to be updated at the client system in response to the client system initiating a request for the web content. The cookie evaluator may be configured to compare a value of the code to the default value. The cached status detector may be configured to use a result of the comparing to determine a cached status of the web content, the cached status to indicate whether the web content has been cached by the client system.

20 Claims, 4 Drawing Sheets



(51) **Int. Cl.**

G06F 17/30	(2006.01)
G06F 12/08	(2016.01)
H04L 29/08	(2006.01)

(56) **References Cited**

U.S. PATENT DOCUMENTS

6,094,662	A	7/2000	Hawes	
6,219,676	B1	4/2001	Reiner	
6,314,463	B1	11/2001	Abbott et al.	
6,366,947	B1	4/2002	Kavner	
6,385,701	B1 *	5/2002	Krein	G06F 12/0815 707/999.008
6,507,854	B1	1/2003	Dunsmoir et al.	
6,625,647	B1	9/2003	Barrick, Jr. et al.	
6,799,214	B1	9/2004	Li	
6,931,439	B1	8/2005	Hanmann et al.	
7,047,281	B1	5/2006	Kausik	
7,269,784	B1	9/2007	Kasriel et al.	
7,330,887	B1	2/2008	Dharmadhikari	
7,426,534	B2	9/2008	Challenger et al.	
7,509,404	B2	3/2009	Agrawal et al.	
8,745,164	B2	6/2014	Choi et al.	
2001/0021962	A1 *	9/2001	Lee	711/118
2002/0007393	A1 *	1/2002	Hamel	709/203
2002/0007413	A1 *	1/2002	Garcia-Luna-Aceves et al.	709/229
2002/0099818	A1	7/2002	Russell et al.	
2003/0065743	A1	4/2003	Jenny et al.	
2003/0101234	A1	5/2003	McBrearty et al.	
2003/0120752	A1	6/2003	Corcoran	
2003/0182357	A1	9/2003	Chess et al.	
2004/0249709	A1	12/2004	Donovan et al.	
2005/0044321	A1	2/2005	Bialkowski et al.	
2006/0064467	A1	3/2006	Libby	
2006/0101341	A1	5/2006	Kelly et al.	
2006/0106807	A1	5/2006	DeVitis et al.	
2006/0149807	A1	7/2006	Ding et al.	
2007/0143344	A1	6/2007	Luniewski et al.	
2007/0150822	A1	6/2007	Mansour et al.	
2008/0155056	A1	6/2008	Zimowski	

FOREIGN PATENT DOCUMENTS

WO	WO-2006055769	A2	5/2006
WO	WO-2009014659	A1	1/2009

OTHER PUBLICATIONS

"U.S. Appl. No. 12/176,253 , Appeal Brief filed Oct. 9, 2012", 18 pgs.
"U.S. Appl. No. 12/176,253 , Response filed Jan. 11, 2012 to Non Final Office Action mailed Oct. 11, 2011", 11 pgs.
"U.S. Appl. No. 12/176,253 , Response filed May 7, 2013 to Non Final Office Action mailed Jan. 7, 2013", 11 pgs.
"U.S. Appl. No. 12/176,253 , Response filed Aug. 9, 2012 to Final Office Action mailed Apr. 9, 2012", 12 pgs.
"U.S. Appl. No. 12/176,253 , Response filed Sep. 25, 2012 to Final Office Action mailed Apr. 9, 2012", 12 pgs.
"U.S. Appl. No. 12/176,253 Non-Final Office Action mailed Sep. 24, 2010", 13 pgs.
"U.S. Appl. No. 12/176,253, Advisory Action mailed Jul. 31, 2013", 3 pgs.
"U.S. Appl. No. 12/176,253, Advisory Action mailed Aug. 23, 2012", 3 pgs.
"U.S. Appl. No. 12/176,253, Appeal Brief filed Oct. 22, 2013", 16 pgs.
"U.S. Appl. No. 12/176,253, Final Office Action mailed Apr. 9, 2012", 15 pgs.
"U.S. Appl. No. 12/176,253, Final Office Action mailed Apr. 19, 2011", 14 pgs.
"U.S. Appl. No. 12/176,253, Final Office Action mailed May 22, 2013", 12 pgs.
"U.S. Appl. No. 12/176,253, Non Final Office Action mailed Jan. 7, 2013", 13 pgs.
"U.S. Appl. No. 12/176,253, Non Final Office Action mailed Oct. 11, 2011", 14 pgs.
"U.S. Appl. No. 12/176,253, Notice of Allowance mailed Jan. 27, 2014", 14 pgs.
"U.S. Appl. No. 12/176,253, Response filed Jan. 24, 2011 to Non-Final Office Action mailed Sep. 24, 2010", 10 pgs.
"U.S. Appl. No. 12/176,253, Response filed Jul. 19, 2011 to Final Office Action mailed Apr. 19, 2011", 10 pgs.
"U.S. Appl. No. 12/176,253, Response filed Jul. 22, 2013 to Final Office Action mailed May 22, 2013", 13 pgs.
"International Application Serial No. PCT/US2008/008825, Search Report mailed Nov. 14, 2008", 4 pgs.
"International Application Serial No. PCT/US2008/008825, Written Opinion mailed Nov. 14, 2008.", 4 pgs.
"WebPage Size Calculator 1.0", [Online]. Retrieved from the Internet: <URL: http://www.topshareware.com/WebPage-Size-Calculator-download-51331.htm>, (Apr. 9, 2007), 2 pgs.

* cited by examiner

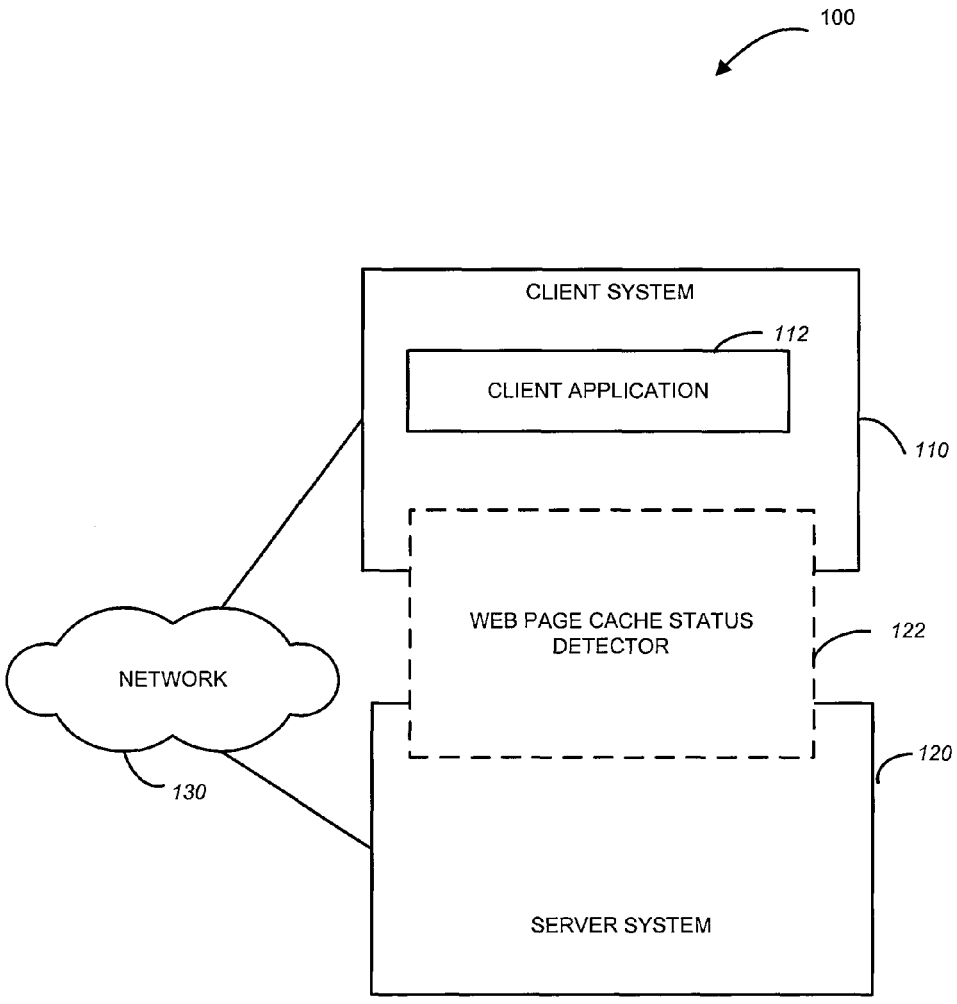

100
CLIENT SYSTEM
112
CLIENT APPLICATION
110
WEB PAGE CACHE STATUS DETECTOR
122
NETWORK
130
120
SERVER SYSTEM

FIG. 1

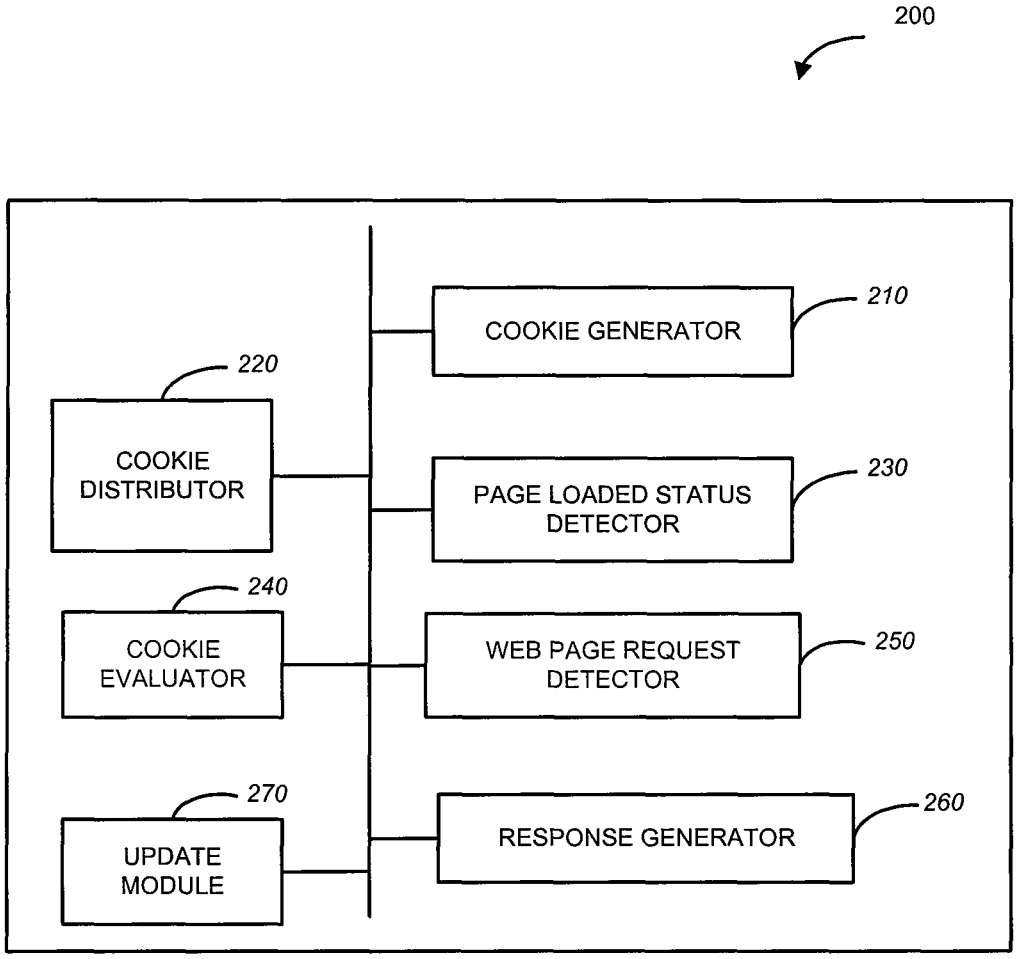

200
COOKIE GENERATOR
210
220
COOKIE DISTRIBUTOR
PAGE LOADED STATUS DETECTOR
230
240
COOKIE EVALUATOR
WEB PAGE REQUEST DETECTOR
250
270
UPDATE MODULE
RESPONSE GENERATOR
260

FIG. 2

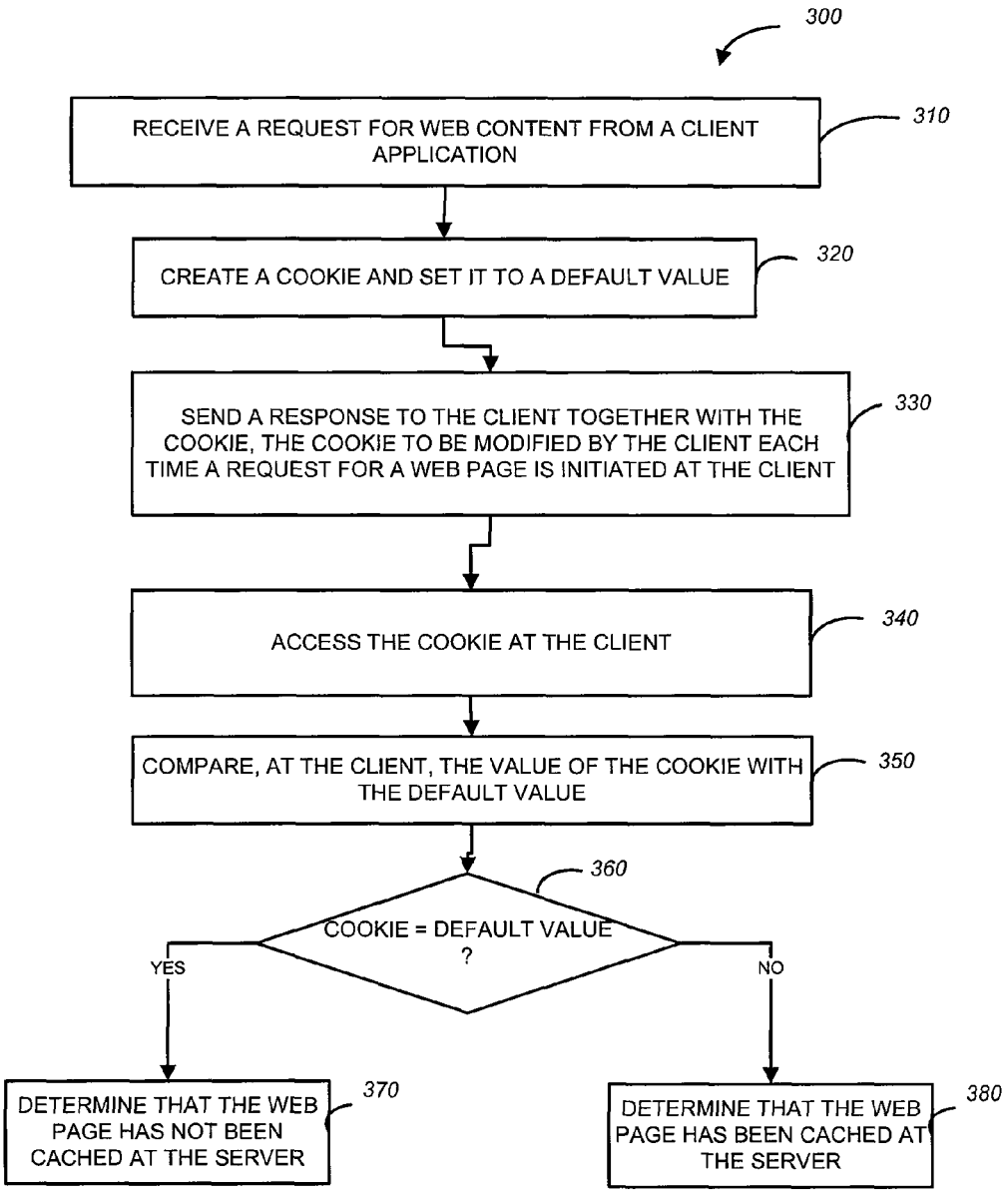

300
RECEIVE A REQUEST FOR WEB CONTENT FROM A CLIENT APPLICATION
310
CREATE A COOKIE AND SET IT TO A DEFAULT VALUE
320
SEND A RESPONSE TO THE CLIENT TOGETHER WITH THE COOKIE, THE COOKIE TO BE MODIFIED BY THE CLIENT EACH TIME A REQUEST FOR A WEB PAGE IS INITIATED AT THE CLIENT
330
ACCESS THE COOKIE AT THE CLIENT
340
COMPARE, AT THE CLIENT, THE VALUE OF THE COOKIE WITH THE DEFAULT VALUE
350
360
COOKIE = DEFAULT VALUE ?
YES
NO
DETERMINE THAT THE WEB PAGE HAS NOT BEEN CACHED AT THE SERVER
370
DETERMINE THAT THE WEB PAGE HAS BEEN CACHED AT THE SERVER
380

FIG. 3

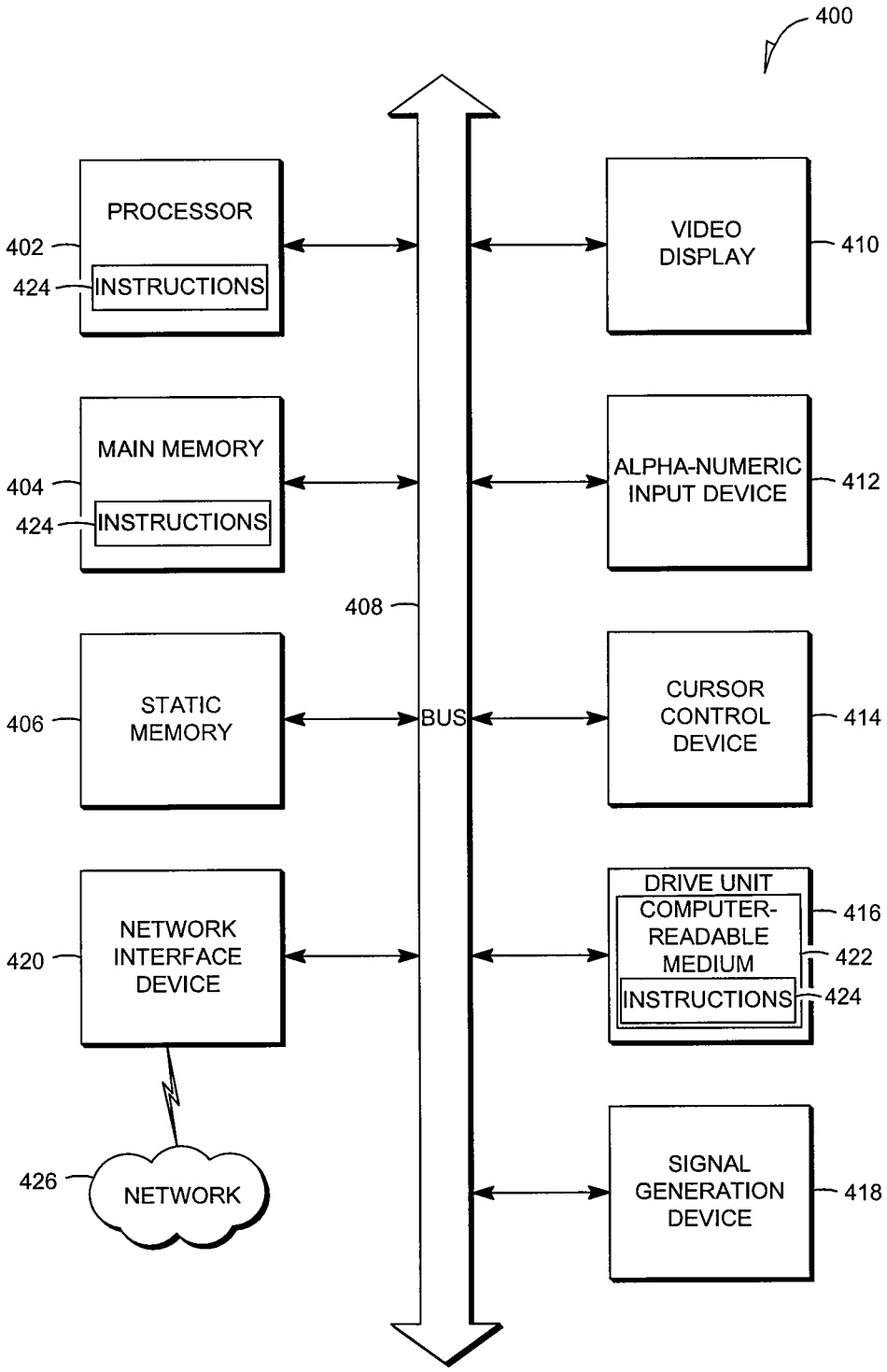

400
PROCESSOR
402
INSTRUCTIONS
424
MAIN MEMORY
404
INSTRUCTIONS
424
STATIC MEMORY
406
NETWORK INTERFACE DEVICE
420
NETWORK
426
408
BUS
VIDEO DISPLAY
410
ALPHA-NUMERIC INPUT DEVICE
412
CURSOR CONTROL DEVICE
414
DRIVE UNIT
COMPUTER-READABLE MEDIUM
INSTRUCTIONS
416
422
424
SIGNAL GENERATION DEVICE
418

FIGURE 4

METHOD AND SYSTEM TO DETECT A CACHED WEB PAGE

RELATED APPLICATIONS

This application is a continuation of U.S. patent application Ser. No. 12/176,253 filed on Aug. 11, 2008, and is related to and hereby claims the priority benefit of U.S. Provisional Patent Application No. 60/950,774 filed Jul. 19, 2007, which applications are incorporated herein by reference in their entirety.

TECHNICAL FIELD

This application relates to a method and system to detect whether a web page has been cached.

BACKGROUND

In the context of web application development and testing, it may be desirable to determine how fast can a web page be loaded in response to a user's request. For example a method for testing web-based applications may include measuring the response time of one or more web pages. Specifically, after the loading of a web page is initiated, an event is received indicating preparation to navigate to the web page and a timer mechanism is started. Another event is received indicating that the web page has completed loading and the timer mechanism is stopped and the elapsed time for the web page to load is determined by accessing the timer readings. This method does not distinguish between loading a web page for the first time and loading a web page that was previously cached at the client system associated with the requesting user. A web page sent from the server computer (server) typically behaves in the same manner as a cached web page does.

BRIEF DESCRIPTION OF DRAWINGS

Embodiments of the present invention are illustrated by way of example and not limitation in the figures of the accompanying drawings, in which like references indicate similar elements and in which:

FIG. **1** is a block diagram showing a network environment within which a method and system to detect whether a web page has been cached may be implemented;

FIG. **2** is a block diagram illustrating a system to detect whether a web page has been cached, in accordance with an example embodiment;

FIG. **3** is a flow chart illustrating a method to detect whether a web page has been cached, in accordance with an example embodiment; and

FIG. **4** illustrates a diagrammatic representation of an example machine in the form of a computer system within which a set of instructions, for causing the machine to perform any one or more of the methodologies discussed herein, may be executed.

DETAILED DESCRIPTION

A method and system to detect whether a web page has been cached is described. In the following description, for purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of an embodiment of the present invention. It will be evident, however, to one skilled in the art that the present invention may be practiced without these specific details.

In one example scenario, in order to evaluate network latency, an approach may be pursued where the network latency is reflected in a value associated with the time duration it takes for a web page or any web content to load in response to a request. Example operations that may be utilized to perform this task are listed below.

1. Upon request, take the current time of the request (t1), and insert this time stamp as apart of the generated web page.
2. On the browser, take a time stamp at the beginning of the rendering of the web page (c1), and at the end of the rendering of the web page (c2).
3. After the page loads, send to the server, e.g., through an image tag, the values of t1 and (c2−c1).
4. On the server, record the new current time (t2) and subtract the (t2) from (t1) to get the end to end time. Then perform a calculation as follows (t2−t1)−(c2−c1), which reflects network latency.

The approach described above may be beneficial in cases of normal web page execution. When a page that is being served is a cached web page, (e.g., the web page is being served in response to a user activating the "back" control button on the browser), t1 represents the cached t1 time. This would affect the result in the end to end result calculation.

In order to determine where a page that has been loaded is a cached web page, an approached has been provided that uses web cookies, which is described below. Hypertext Transfer Protocol (HTTP) cookies, referred to as web cookies or merely cookies, are server generated identifiers stored on the computer of the person browsing the web, which are sent to the server with each request. In one example embodiment, a specified cookie (e.g., a code) may be set on the server to a default value for every request to access a particular web page. On the client, for every request to access the web page, the default value of the specified cookie is being modified. On the client, if the read cookie value is different from the default value stored on the server, it is concluded that the web page never hit the server, and therefore has been cached. Another way to describe this approach is as follows.

On the server for every request, we set a specified cookie to its default value and on the client we modify this value. On the client, if the read cookie value is not the default value, we know the page never hit the server, and therefore has been cached.

In one example embodiment, the method and system to determine whether a web page has been cached may be utilized as described below. Suppose an advertisement from a 3rd party is served up on a web given page. For every request for the web page, a unique identifier (e.g., generated on the client) may be added to the query string of the call in order to ensure that the advertisement call (ad request) is not cached. The result is that, even on cached web pages, the new advertisement call would have a new identifier, making it appear as if a new request has been made. which affects metrics. With the above solution, in one example embodiment, the identifier may be saved in the cookie. New requests would wipe out this value. If it is determined that this value is present, the cached identifier may be used with the ad request so that the server side could identify which calls are new and which calls are cached.

Example system to detect whether a web page has been cached may be described with reference to a network environment **100** illustrated in FIG. **1**. The network environment **100** may include a client system (or client) **110** and a server system (or server) **120**. The client system **110** and the server system **120** may be in communications with each other via a network **130**. The communications network **130**

may be a public network (e.g., the Internet, a wireless network, a public switched telephone network (PSTN), etc.) or a private network (e.g., LAN, WAN, Intranet, etc.). Also shown in FIG. **1** is a web page cache status detector **122**. The web page cache status detector **122** may reside at the server system **120**, at the client system **110**, or be distributed between the server system **120** and the client system **110**. The client system **110** is shown to host a client application (e.g., a web browser application) **112**. The web page cache status detector **122** may be utilized to determine whether a web page that is a subject of information being gathered at the server system **120** has been cached by the client system **110**. Example embodiment of a system to detect whether a web page has been cached may be described with reference to FIG. **2**.

FIG. **2** is a block diagram illustrating a system **200** to determine whether a web page has been cached, in an example embodiment of the web page cache status detector **122** shown in FIG. **2**. The system **200** comprises a cookie generator **210** to generate HTML cookies, the cookie distributor **220** to store cookies at one or more client systems, and a so-called page loaded status detector **230** that determines whether a web page has been loaded at the client, and a cookie evaluator **240** to use cookies to determine whether a web page that has been loaded at the client was loaded from the server or from the client's cache. The system **200** further comprises a web page request detector **250** to receive a request for a web page from a client, and a response generator **260** to provide the requested web page to the client together with an instruction to update the value of the cookie stored at the client to the default value. In some embodiments, a default value of the cookie may be updated by an update module **270**. Example operations performed by the web page cache status detector **122** may be discussed with reference to FIG. **3**

FIG. **3** is a flow chart of a method **300** to determine whether a web page has been cached, according to one example embodiment. The method **300** may be performed by processing logic that may comprise hardware (e.g., dedicated logic, programmable logic, microcode, etc.), software (such as run on a general purpose computer system or a dedicated machine), or a combination of both. In one example embodiment, the processing logic resides at the server system **140** of FIG. **1** and, specifically, at the system **200** shown in FIG. **2**.

As shown in FIG. **3**, the method **300** commences at operation **310**, where the web page request detector **250** of FIG. **2** received a request for web content from a client application. At operation **320**, the cookie generator **210** of FIG. **2** creates a cookie at a server system and sets the cookie to a default value. The method **300** may utilize the cookie distributor **220** of FIG. **2** and the response generator **260** of FIG. **2** send the response to the request for the web content and to store the cookie at the client system, at operation **330**. The stored cookie is to be modified at the client system each time a request for a web page is initiated at the client. There are numerous ways in which the value of the cookie can be modified. For example, the value may be incrementally increased, decreased, or set to a randomly generated value.

At operation **340**, the cookie is accessed at the client system. At operation **350**, the cookie evaluator **240** of FIG. **2**, compares the value of the cookie received with the response to the request for the web content. If it is determined, at operation **380**, that the two values match, the method **300** determines, at operation **370**, that the web page has not been cached. Such determination may be made because, as described above, while each time a request for the web page is initiated, the value of the cookie at the client is updated, each time the web page is provided to the client from the server, the value of the cookie at the client is updated to a predetermined default value. If it is determined, at operation **360**, that the two values are distinct from each other, the method **300** determines, at operation **380**, that the web page has been cached, because it indicates that while the web page has been loaded, the client did not receive an instruction to update the value of the cookie.

FIG. **4** shows a diagrammatic representation of machine in the example form of a computer system **400** within which a set of instructions, for causing the machine to perform any one or more of the methodologies discussed herein, may be executed. In alternative embodiments, the machine operates as a standalone device or may be connected (e.g., networked) to other machines. In a networked deployment, the machine may operate in the capacity of a server or a client machine in server-client network environment, or as a peer machine in a peer-to-peer (or distributed) network environment. The machine may be a personal computer (PC), a tablet PC, a set-top box (STB), a Personal Digital Assistant (PDA), a cellular telephone, a portable music player (e.g., a portable hard drive audio device such as an MP3 player), a web appliance, a network router, switch or bridge, or any machine capable of executing a set of instructions (sequential or otherwise) that specify actions to be taken by that machine. Further, while only a single machine is illustrated, the term "machine" shall also be taken to include any collection of machines that individually or jointly execute a set (or multiple sets) of instructions to perform any one or more of the methodologies discussed herein.

The example computer system **400** includes a processor **402** (e.g., a central processing unit (CPU), a graphics processing unit (GPU) or both), a main memory **404** and a static memory **406**, which communicate with each other via a bus **408**. The computer system **400** may further include a video display unit **440** (e.g., a liquid crystal display (LCD) or a cathode ray tube (CRT)). The computer system **400** also includes an alphanumeric input device **442** (e.g., a keyboard), a user interface (UI) navigation device **444** (e.g., a mouse), a disk drive unit **446**, a signal generation device **448** (e.g., a speaker) and a network interface device **420**.

The disk drive unit **446** includes a machine-readable medium **422** on which is stored one or more sets of instructions and data structures (e.g., software **424**) embodying or utilized by any one or more of the methodologies or functions described herein. The software **424** may also reside, completely or at least partially, within the main memory **404** and/or within the processor **402** during execution thereof by the computer system **400**, the main memory **404** and the processor **402** also constituting machine-readable media.

The software **424** may further be transmitted or received over a network **426** via the network interface device **420** utilizing any one of a number of well-known transfer protocols (e.g., HTTP).

While the machine-readable medium **422** is shown in an example embodiment to be a single medium, the term "machine-readable medium" should be taken to include a single medium or multiple media (e.g., a centralized or distributed database, and/or associated caches and servers) that store the one or more sets of instructions. The term "machine-readable medium" shall also be taken to include any medium that is capable of storing, encoding or carrying a set of instructions for execution by the machine and that cause the machine to perform any one or more of the methodologies of the present invention, or that is capable of storing, encoding or carrying data structures utilized by or

associated with such a set of instructions. The term "machine-readable medium" shall accordingly be taken to include, but not be limited to, solid-state memories, optical and magnetic media, and carrier wave signals. Such medium may also include, without limitation, hard disks, floppy disks, flash memory cards, digital video disks, random access memory (RAMs), read only memory (ROMs), and the like.

The embodiments described herein may be implemented in an operating environment comprising software installed on a computer, in hardware, or in a combination of software and hardware.

Although embodiments have been described with reference to specific example embodiments, it will be evident that various modifications and changes may be made to these embodiments without departing from the broader spirit and scope of the invention. Accordingly, the specification and drawings are to be regarded in an illustrative rather than a restrictive sense.

What is claimed is:

1. A computer-implemented system comprising:

one or more processors provided at a server computer system;

a web page request detector, implemented using the one or more processors, to receive a request for web content and a client code from a client application executing at a client system, the client code includes an identifier associated with the requested web content;

a response generator, implemented using the one or more processors, to provide the web content to the client system, in response to the request from the client application, together with a request to set the client code at the client system to an uncached value;

an evaluator, implemented using the one or more processors, to compare the client code to the uncached value in response to detecting that the web content has been loaded at the client system; and

a cached status detector, implemented using the one or more processors, to determine that the web content has been cached by the client system, based on determining that the client code is distinct from the uncached value.

2. The system of claim **1**, wherein: the web content includes third party content associated with the identifier; the request to set the client code to the uncached value includes setting the client code to the identifier;

the evaluator comparing the client code to the uncached value includes comparing the client code to the identifier in response to receiving, at the server computer system, a request for the third party content and the identifier; and

the cached status detector determining that the web content has been cached includes determining whether the request for the third party content is a request for new content based on determining that the client code is distinct from the identifier.

3. The system of claim **2**, wherein the cached status detector is to determine that the request for the third party content is a new request based on the result of comparing of the value of the identifier stored in the client code at the client system and a previously stored third party content identifier being a match.

4. The system of claim **2**, wherein the cached status detector is to determine that the request for the third party content is not a new request based on the result of comparing of the value of the identifier stored in the client code at the client system and a previously stored third party content identifier being a non-match.

5. The system of claim **2**, wherein the third party content is an advertisement.

6. The system of claim **1**, wherein the client code is to be modified at the client system in response to the client system initiating a request for the web content.

7. The system of claim **1**, wherein the cached status detector is to determine that the web content has not been cached by the client system, based on determining that the value of the client code is the same as the uncached value.

8. The system of claim **1**, comprising a code generator, implemented using one or more processors, to set a code to the uncached value at a server system.

9. The system of claim **1**, comprising a code distributor, implemented using one or more processors, to provide the code to a client system as the client code.

10. The system of claim **1**, wherein the web content is a web page.

11. A method comprising operations performed at a server computer system:

receiving a request for web content and a client code from a client application executing at a client system, the client code includes an identifier associated with the requested web content;

providing the web content to the client system, in response to the request from the client application, together with a request to set the client code at the client system to an uncached value;

comparing the client code to the uncached value in response to detecting that the web content has been loaded at the client system; and

determining, using one or more processors, that the web content has been cached by the client system, based on determining that the client code is distinct from the uncached value.

12. The method of claim **11**, wherein: the web content includes third party content associated with the identifier; the request to set the client code to the uncached value includes setting the client code to the identifier; the comparing the client code to the uncached value includes comparing the client code to the identifier in response to receiving, at the server system, a request for the third party content and the identifier; and the determining that the web content has been cached includes determining whether the request for the third party content is a request for new content based on determining that the client code is distinct from the identifier.

13. The method of claim **12**, wherein the determining that the request for the third party content is a new request is based on the result of comparing of the value of the identifier stored in the client code at the client system and a previously stored third party content identifier being a match.

14. The method of claim **12**, wherein the determining that the request for the third party content is not a new request based on the result of comparing of the value of the identifier stored in the client code at the client system and a previously stored third party content identifier being a non-match.

15. The method of claim **12**, wherein the third party content is an advertisement.

16. The method of claim **11**, comprising modifying the value of the client code in response to the client system initiating a request for the web content.

17. The method of claim **11**; wherein the determining that the web content has not been cached by the client system, based on determining that the value of the client code is the same as the uncached value.

18. The method of claim **11**, comprising setting a code to the uncached value at a server system.

19. The method of claim 11, comprising providing the code to a client system as the client code.

20. A machine-readable non-transitory storage medium having instruction data executable by a machine to cause the machine to perform operations comprising:

receiving a request for web content and a client code from a client application executing at a client system; the client code includes an identifier associated with the requested web content;

providing the web content to the client system, in response to the request from the silent application; together with a request to set the client code at the spent system to an uncached value;

comparing, at the server computer system; a value of the client code to the uncached value in response to detecting that the web content has been loaded at the client system; and

determining that the web content has been cached by the client system, based on determining that the client code is distinct from the uncached value.

* * * * *